Gestione degli eventi in Java

package java.awt.event.*

Il modello degli eventi

- I componenti dell'awt generano eventi in seguito alle azioni dell'utente
  - movimento del mouse
  - click
  - pressione di un tasto
  - etc.
- Gli eventi vengono passati ad una lista di *ascoltatori* associata al componente

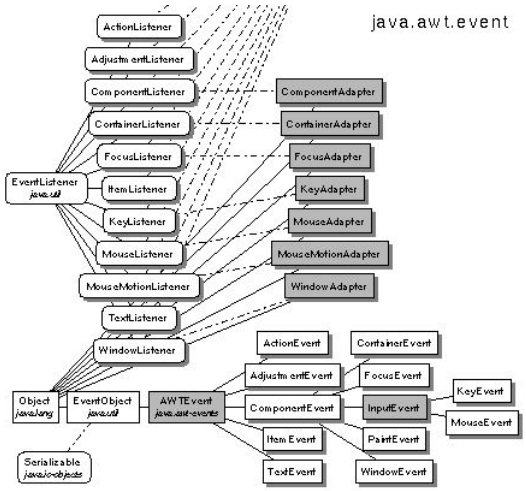
Ascoltatori (*Listener*)

- Al momento della creazione di un nuovo componente awt, questi ha la lista degli ascoltatori associati vuota
- Dopo che si sono aggiunti degli ascoltatori alla lista del componente, ogni volta che viene generato un evento, viene passato agli ascoltatori che, eventualmente, eseguiranno del codice
- È come se il componente fosse un uomo che in seguito ad un incendio (=evento) grida “al fuoco” (=lancia un evento) e poi non fa nulla, mentre il vigile del fuoco in ascolto (=ascoltatore) spegne l’incendio

Ascoltatori 2

- Ad un componente si possono associare più ascoltatori per lo stesso evento (=ci possono essere più vigili del fuoco pronti a rispondere) ed anche ascoltatori per eventi diversi (=posso associare un vigile per gli incendi ed un poliziotto per i furti)
- Un ascoltatore può essere associato a più componenti contemporaneamente (=un vigile del fuoco risponde alle chiamate di più persone)

Gerarchia di classi java.awt.event.*



Come si chiudono le finestre

```
import java.awt.*;
import java.awt.event.*;
class finestra {
  public static void main(String[] args) {
    Frame f = new Frame("titolo");
    f.setBounds(20,20,200,150);
    f.addWindowListener(new ascoltatore());
    f.setVisible(true);
  }
}
```

```
class ascoltatore implements WindowListener {
  public void windowClosing(WindowEvent e) {
    e.getWindow().dispose();
  }
  public void windowClosed(WindowEvent e) {
    System.exit(0);
  }
  public void windowOpened(WindowEvent e) {}
  public void windowIconified(WindowEvent e) {}
  public void windowDeiconified(WindowEvent e) {}
  public void windowActivated(WindowEvent e) {}
  public void windowDeactivated(WindowEvent e) {}
}
```

Come si chiudono le finestre 2

- **`addWindowListener(WindowListener l)`** è il metodo della classe **`Window`** per aggiungere un ascoltatore degli eventi relativi alle finestre
- **`windowClosing`** viene eseguito quando si fa click sul bottone in alto a destra della finestra
- **`e.getWindow()`** restituisce la finestra che ha generato l'evento
- **`dispose()`** chiude la finestra
- **`windowClosed`** viene eseguito quando la finestra è chiusa

Ascoltatori vs. Adattatori

- Se si vuole scrivere una classe che gestisce un tipo di evento awt, dobbiamo farle *implementare* l'interfaccia dell'opportuno Listener
  - quindi dobbiamo scrivere una classe che implementi TUTTI i metodi dell'interfaccia
- In alcuni casi abbiamo a disposizione delle classi astratte (adattatori) che implementano già tutti i metodi

Ascoltatori vs. Adattatori 2

- Invece di scrivere una classe ed implementare tutti i metodi dell'interfaccia possiamo *ereditare* i metodi da un adattatore (estendendo la classe) e poi sovrascrivere solo quello che ci interessa
- **CONTRO**: in java non c'è ereditarietà multipla, quindi se una classe vuole gestire diversi tipi di eventi non si può farle ereditare i metodi da diversi adattatori. In quel caso occorre implementare le interfacce

Chiudere le finestre con adattatore

```
import java.awt.*;
import java.awt.event.*;
class finestra2 {
  public static void main(String[] args) {
    Frame f = new Frame(“titolo”);
    f.setBounds(20,20,200,150);
    f.addWindowListener(new adattatore());
    f.setVisible(true);
  }
}
class adattatore extends WindowAdapter {
  public void windowClosing(WindowEvent e) {
    e.getWindow().dispose();
  }
  public void windowClosed(WindowEvent e) {
    System.exit(0);
  }
}
```

Azioni e ascoltatori di azioni

```
import java.awt.*;
import java.awt.event.*;
class finestra3 {
   public static void main(String[] args) {
       Frame f = new Frame(“titolo”);
       Panel p = new Panel();
       Button b = new Button(“chiudi”);
       b.addActionListener(new terminatore(f));
       f.add(p); p.add(b);
       f.setBounds(20,20,200,150);
       f.setVisible(true);
   }
}
```

Azioni e ascoltatori di azioni 2

```
class terminatore implements ActionListener {
  Frame fin;
  terminatore(Frame fin) {
    this.fin = fin;
  }
  public void actionPerformed(ActionEvent e) {
    fin.dispose();
    System.exit(0);
  }
}
```

Azioni e ascoltatori di azioni 3

- **`addActionListener(ActionListener l)`** è il metodo delle classi **`Button`**, **`List`**, **`MenuItem`** e **`TextField`** per aggiungere un ascoltatore di eventi
- L'interfaccia **`ActionListener`** contiene il solo metodo **`actionPerformed`** e quindi non esiste un adattatore

Eventi del mouse

```
import java.awt.*;
import java.awt.event.*;
class finestra4 {
   public static void main(String[] args) {
       Frame f = new Frame(“titolo”);
       Panel p = new Panel();
       Button b = new Button(“ciao”);
       b.addMouseListener(new ascoltatopo());
       f.add(p); p.add(b);
       f.setBounds(20,20,200,150);
       f.setVisible(true);
   }
}
```

Eventi del mouse 2

```
class ascoltatopo implements MouseListener {
  public void mouseClicked(MouseEvent e) {
    System.out.println(“click”);    }
  public void mousePressed(MouseEvent e) {
    System.out.println(“premuto”);  }
  public void mouseReleased(MouseEvent e) {
    System.out.println(“lasciato”); }
  public void mouseEntered(MouseEvent e) {
    System.out.println(“sopra”);    }
  public void mouseExited(MouseEvent e) {
    System.out.println(“fuori”);    }
}
```

Eventi del mouse 3

- **`addMouseListener(MouseListener l)`** è un metodo di **`Component`** (e quindi di tutte le sue sottoclassi)
- Esiste un adattatore **`MouseAdapter`**

Eventi per il testo

```
import java.awt.*;
import java.awt.event.*;
class finestra5 {
  public static void main(String[] args) {
    Frame f = new Frame("titolo");
    Panel p = new Panel();
    TextField t = new TextField(20);
    t.addTextListener(new ascoltatesto());
    f.add(p); p.add(t);
    f.setBounds(20,20,200,150);
    f.setVisible(true);
  }
}
```

Eventi per il testo 2

```
class ascoltatesto implements TextListener {
  public void textValueChanged(TextEvent e) {
    System.out.print("cambiato! in ");
    TextComponent txt = (TextComponent)e.getSource();
    System.out.println(txt.getText());
  }
}
```

Eventi per il testo 3

- **`addTextListener(TextListener l)`** è un metodo di **`TextComponent`** (e quindi di tutte le sue sottoclassi)
- Non esiste un adattatore (infatti l'interfaccia contiene un solo metodo)
- **`getSource()`** restituisce l'**`Object`** che ha generato l'evento. Per poter chiamare i metodi della classe **`TextComponent`**, (**`getText()`**) occorre prima fare un *cast*

Selezionare elementi

```
import java.awt.*; import java.awt.event.*;
class finestra6 {
  public static void main(String[] args) {
    Frame f = new Frame("titolo"); Panel p = new Panel();
    Choice l = new Choice();
    l.add("html 4.0"); l.add("xhtml 1.0"); l.add("css 1");
    l.add("css 2"); l.add("xhtml 1.1"); l.add("html 4.01");
    l.addItemListener(new ascSelezione());
    f.add(p); p.add(l); f.setBounds(20,20,200,150);
    f.setVisible(true);
  }
}
class ascSelezione implements ItemListener {
  public void itemStateChanged(ItemEvent e) {
    System.out.println("Selezionato "+e.getItem());
  }
}
```

Selezionare elementi 2

- **`addItemListener(ItemListener l)`** è un metodo delle classi **`Choice`** e **`List`**
- Non esiste un adattatore (infatti l'interfaccia contiene un solo metodo)
- **`getItem()`** restituisce l'**`Object`** che è stato interessato dall'evento (non restituisce l'oggetto che ha generato l'evento che è la lista, ma l'elemento)